1. Insert, in order, the elements 2, 19, 15, 3, 42, 11, 9, 18, 12, 5 into a 2 − 3 − 4 tree.

Convert the 2 − 3 − 4 tree you constructed into a red-black binary search tree. Indicate the color of each node by using letters B (black) and R (red). You may omit representing the NIL nodes.

Diagram

Description automatically generated

**Converting to a Red-Black Tree:**

* (3, 11, 15) is a 4-node that splits. 20 becomes the black parent. 11 and 15 become red children.
* (2) is a 2-node that becomes a black child of 3.
* (5, 9) – is a 3-node that splits into one red and one black node. 5 will be the red node while 9 becomes the black parent.
* (12)- 2 node that becomes a black child of 15
* (18, 19, 42) – 4-node that splits. 19 becomes the black parent. 18 and 42 becomes the red children of 19. 19 is also the child node of red 15.

Chart, bubble chart

Description automatically generated

1. Insert in order the keys 2, 19, 15, 3, 42, 11, 9, 18, 12, 5 into a red-black BST tree. Show your tree after every insertion. Indicate the color of each node by letters B (black) and R (red). You may omit representing the NIL nodes.

After that, convert the red-black BST you have constructed into a 2 − 3 − 4 tree, using the steps we discussed in class.

**Red-Black tree**:

Inserting 2: Root will be black

A picture containing toilet

Description automatically generated

Inserting 19: To maintain equal black height, the node will be red.

Diagram, schematic

Description automatically generated

Inserting 15: Same step as 19. Black height property preserved.

Diagram, schematic

Description automatically generated

Inserting 3: Inserting 3 as the left child of 15. Case 1 violation: Red-Red violation. Solved by recoloring 19 and 15 to become black nodes.

Bubble chart

Description automatically generated

Inserting 42: Inserting 42 as the right child of 15. Red-Black property maintained.

Bubble chart

Description automatically generated

Inserting 11: Inserting 11 as the right child of 3. This becomes a case 2 Red-Red violation: Will rotate the sub-tree left to transform it into case 3.

* Case 3 violation: Sibling of the 19 parent is also black. Must right rotate and recolor tomaake 19 the black parent of red children 3 and 19.

Diagram, schematic

Description automatically generated

Inserting 9: Must insert 9 to be the left child of 15. Red-Black property maintained.

Diagram, schematic

Description automatically generated

Inserting 18: Insert 18 as the left child of 42. Case 1 violation: Sibling of parent 42 is a red node. Recoloring 42 and 9 to be black while grandparent 15 becomes red.

A picture containing text, device

Description automatically generated

Inserting 12: Inserting 12 to be the right child of 9. No Red-Red violation found.

A picture containing text, device, stationary

Description automatically generated

Inserting 5: Inserting 5 to be the left child of 9. No violation found.

A picture containing text, grass, building, device

Description automatically generated

1. (5 points) What is the minimum and maximum height of a 2 − 3 − 4 tree of n nodes. Justify your answer.

* Since 2-3-4 tree can have up to n nodes that hold 3 keys and 4 links, the minimum height would be a tree where all nodes have 4 links and 3 keys.
  + 4 links is the same as holding 4 nodes so the minimum height would be
* The maximum height would be a tree that holds only nodes with 2 links and 1 key which would be a standard binary tree
  + The maximum height would be

1. Show that the longest simple path from a node x in a red-black tree to a descendant leaf has length at most twice that of the shortest simple path from x to a descendant leaf.

In a Red-Black tree, the black-height must be the same number for all paths which means the longest path will be determined by how many red nodes we can add for each black node in one path. This also means that the black-height is the minimum number of nodes we must travel through on a single path.

* Shortest path: All black nodes. Can be at minimum the height of the tree as bh.

The longest path that’s allowed would be a path that alternates between red and black nodes for every step. For the longest path, the number of red nodes and black nodes would be nearly equal, essentially making the height twice that of the black height of that path which gives us 2\*bh.

* Maximum height: Alternating between red nodes and black nodes. This would allow the maximum height to be twice the size of the black-height: 2\*bh.

The shortest path can go as low as bh. The longest path can go as high as 2\*bh which is at most twice that of the shortest path.

1. CLRS page 322, question 13.3-4

13.3-3

Suppose that the black-height of each of the subtrees in Figures 13.5 and 13.6 is k. Label each node in each figure with its black-height to verify that the indicated transformation preserves property 5.

A picture containing sky, different, bunch, sword

Description automatically generated

* – Black-Height of

Figure 13.5

Diagram, schematic

Description automatically generated

Figure 13.6

13.3-4

Professor Teach is concerned that RB-INSERT-FIXUP might set T.nil.color to RED, in which case the test in line 1 would not cause the loop to terminate when is the root. Show that the professor’s concern is unfounded by arguing that RBINSERT-FIXUP never sets T.nil.color to RED.

RB-INSERT-FIXUP(T,z)

1 while z.p.color == RED

2 if z.p = = z.p.p.left

3 y = z.p.p.right

4 if y.color == RED

5 z.p.color = BLACK // case 1

6 y.color = BLACK // case 1

7 z.p.p.color = RED // case 1

8 z = z.p.p // case 1

9 else if z == z.p.right

10 z = z.p // case 2

11 LEFT-ROTATE(T,z) // case 2

12 z.p.color = BLACK // case 3

13 z.p.p.color = RED // case 3

14 RIGHT-ROTATE(T, z.p.p) // case 3

15 else (same as **then** clause with “right” and “left” exchanged)

16 T.root.color = BLACK

Line 7 and line 13 are the two scenarios when a node gets colored red.

* When a node is inserted into the loop, it starts off colored red to begin the recoloring and rotation loop. It checks for what type of case violation is occurring in the Red-Black tree before fixing the issue.
* In line 7, during case 1, z colors its grandparent node to be red.
* In line 13, during case 3, z colors its grandparent node red.

6. When inserting a node into a red-black tree there might occur a red-red violation.

Show how each of the cases (i.e. case 1, case 2, and case 3) to solve the red-red

violation can be understood by showing how it would be solved in a 2 − 3 − 4 tree

(aka 2 − 4 tree)

7. Thank goodness you are in CS6033. You suddenly realize that there was a problem with your previous solution to knowing if someone is a subscriber.... what if someone canceled their subscription? You had no way of removing them from your hash table S without potentially also removing someone else whose email hashed to the same location.

You decide to invest in a cloud service and you now would like to determine if someone is a subscriber in O(log n) worst case time, and enter and remove a person in O(log n) worst case time.

Design an algorithm and justify your algorithm runs in O(log n) time for all operations.

8. You discover you have a talent for public speaking! In fact, you love to be in front of a crowd. Now that covid restrictions are lifting, you decide to hold in-person seminars on how to pick stocks.

To save money on hotel fees, you will find a location a friend or family member lives, and when you go visit them you will hold a seminar in that location. As part of your marketing strategy, you decide to only invite established subscribers within the same geographic region to attend your seminar.

To quickly determine the names of all your subscribers who live in a certain area, design a data structure and implement the following operations:

Specifically:

• Build(T, A) Given an unsorted array of your subscribers,1 create your data structure, T to hold the items in A, in O(n log n).

• Local Subscriber List(T, zipcode) The time it takes to determine the email addresses of all the your subscribers with a specific zip code is O(S + log n) where n is the number of your subscribers. (You can put the names in to an array A.)

• Insert(T, s) Add new subscribers, s, in O(log n) time.

9. Unfortunately your first seminar did not go well. Very few people showed up. You discovered that another market guru held a seminar two days before you held your seminar. Instead of cursing your luck and giving up on your dream, you decide to collaborate. You contact the market guru and the two of you decide to join forces; you will hold joint seminars.

To make this work, you need to combine mailing lists. The market guru will send you a sorted array, A, of their subscribers where it is sorted based on zip codes and then within each zip code it is sorted by name.

Design a data structure that supports the following operations: Combine to insert a list of items into your data structure in O(m1 + m2), Insert(T, s) to insert a new subscriber into your data structure in time O(log(m1 + m2)), and Local Subscriber List to return a list of subscribers within a certain zip code in

O(S + log(m1 + m2)) where S is the number of names with that zip code.

You only need to implement the following method:

• Insert List(T, A) that takes both your mailing lists and creates a new mailing list. The time to combine the market guru’s list with your list 2 is O(m1 + m2) worst case time where m1 is the size of your mailing list and m2 is the size of market guru’s mailing list. Remove duplicates. (You may assume all names are unique in a zip code - i.e. there are not two ”John Smiths” in zip code 10003.) Justify the running time of Insert List.

3

10. Wow. You didn’t know how many other people were also holding seminars. If you had known this, you would have never started holding seminars, instead you would have gone into investment banking.

You and the market guru have now teamed up with the M other speakers. You will create a master list of all the subscribers names:

• Each of the M speakers has provided you their subscriber lists as a sorted array: sorted by zip code and within each zip code ordered alphabetically by the subscriber’s name. Let ni be the number of names in the ith sorted array,

Ai for 1 ≤ i ≤ M.

• You and the marketing guru’s mailing list is in the data structure you used in question 9. The number of names in your list is n0. Design a data structure that supports the following operations:

• Insert Lists(T, A1, A2, . . . , AM) to insert the lists from the other speakers into your data structure in O(n log M) time where n = PM i=0 ni

• Insert(T, s) to insert a new subscriber into your data structure in time O(log(n))

• Local Subscriber List to return a list of subscribers within a certain zip code in O(S + log(n)) where S is the number of names with that zip code You only need to implement the following method:

• Insert Lists(T, A1, A2, . . . , AM) This method add the mailing lists Ai to your data structure in O(n log M) worst time where n = PM i=0 ni and ni = |Ai| is the number of items in the i’th subscriber list.4 Remove duplicates(again, you may assume the names are unique within a zip code).

In your method, clearly write the data structures you will use and what you will store in each data structure. Justify the running time of Insert Lists.

5

Hint: this question heavily uses some of the data structures we covered in previous lectures along with the data structure discussed in lecture 4.

11. (3 bonus points) Think of a good 6 exam question for the material covered in Lecture

4.